Microsoft
.net developer
training

Building ASP®.NET Applications

Agenda: Day 1

Module 01: What is Microsoft® .NET?

Module 02: Visual Studio® .NET

Module 03: The .NET Languages

▶ Module 04: Building ASP®.NET Applications

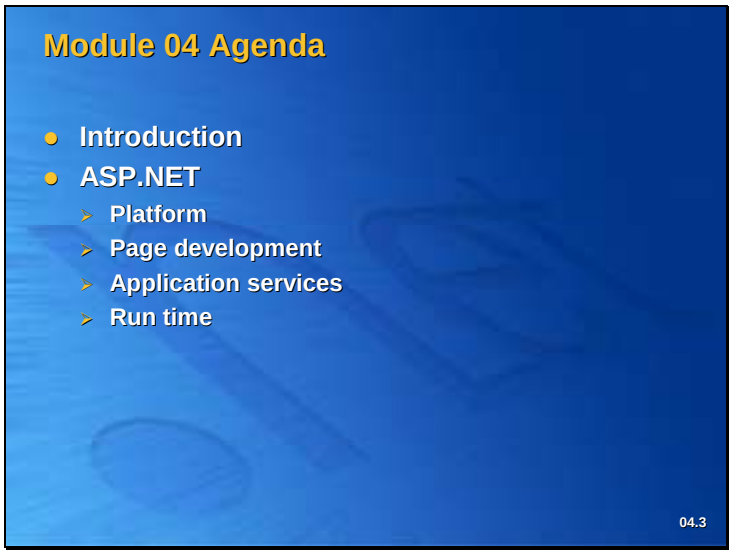
Module 05: Using Data Access in .NET Applications

Module 06: Object Oriented Concepts in .NET

Module 07: Creating Windows Applications

Module 08: Using XML in .NET

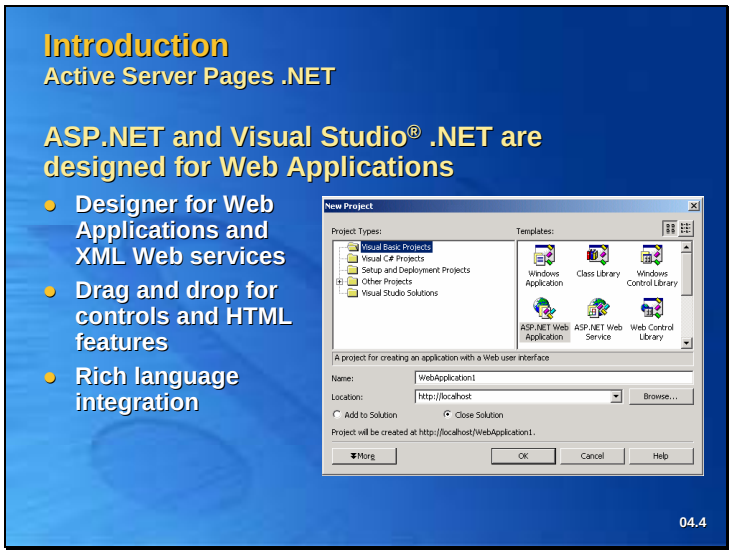
 ©2002 Microsoft Corporation



Module 04 Agenda

What you will learn:

- How to develop Web Applications and pages by using ASP.NET
- What are the application services provided by the ASP.NET Common Language Runtime
- How easy it is to deploy your new Web Applications



Introduction

Active Server Pages .NET

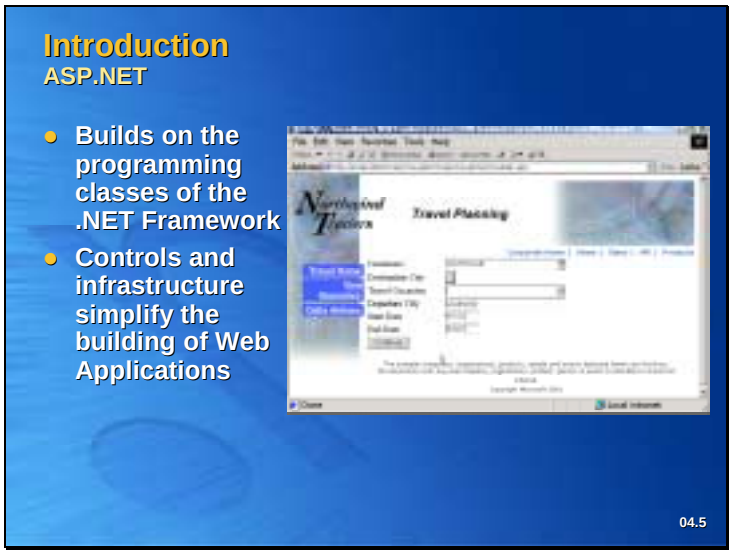
Active Server Pages .NET (ASP.NET) is more than simply the next version of ASP. It is a unified Web development platform that enables developers to build enterprise-class Web Applications. While ASP.NET is largely syntax-compatible with ASP, it provides a new programming model and infrastructure that enables a powerful class of applications. You can augment your existing ASP applications by incrementally adding ASP.NET functionality.

ASP.NET provides a rich development environment that features drag and drop development and a powerful event driven architecture. It is a powerful upgrade to ASP and Web development in general that exists within the .NET Framework.

ASP.NET is a compiled .NET Framework-based environment where you can author applications in any .NET Framework compatible language including Visual Basic®, C# and JScript. Additionally, the .NET Framework platform is available to any ASP.NET application. Developers can easily access the benefits of the .NET Framework, which include a fully-managed, protected and feature-rich application execution environment and simplified development and deployment as well as seamless integration with a variety of languages.

You can write your ASP.NET applications in Notepad if you like, but it is easier to use Visual Studio .NET with its powerful GUI interface and rich set of tools for developing ASP.NET applications. The features that were included in Visual InterDev in the past are an addition of Visual Studio .NET.

 ©2002 Microsoft Corporation



Introduction

ASP.NET

ASP.NET builds on the programming classes of the .NET Framework, providing a Web Application model with a set of controls and infrastructure that make it simple to build Web Applications. It also includes controls that encapsulate common HTML user-interface elements, such as text boxes and drop-down menus. These controls run on the server, however, and push their user interface as HTML to the browser.

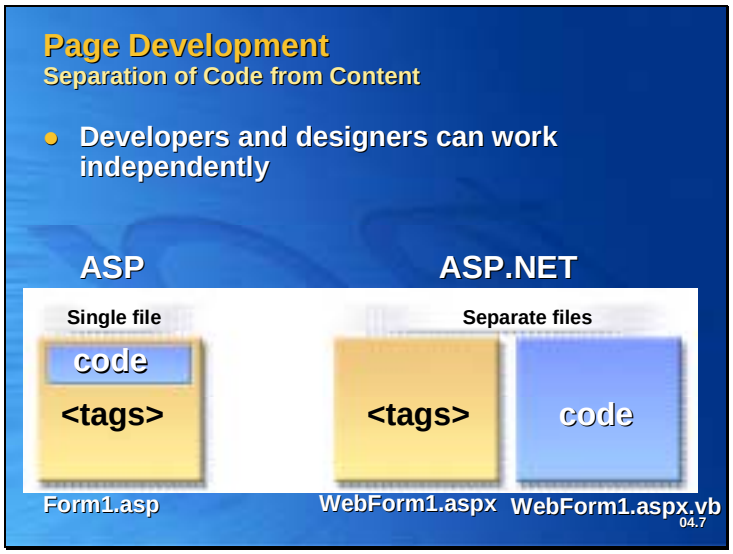
On the server, the controls expose an object-oriented programming model that brings the richness of object-oriented programming to the developer. ASP.NET also provides infrastructure services, such as session state management and process recycling, which further reduce the amount of code a developer must write and increase application reliability. In addition, ASP.NET uses these concepts to enable developers to deliver software as a service. Using Web services features, ASP.NET developers can write their business logic and use the ASP.NET infrastructure to deliver that service by using SOAP.

The resulting HTML from ASP.NET applications is browser neutral, providing a rich architecture for developing HTML 3.2 and higher applications. For instance, the page shown on this slide uses ASP.NET server controls that render standard HTML to the browser.

Page Development Topics

The following slides discuss how to build Web pages by using the new ASP.NET architecture.

 ©2002 Microsoft Corporation



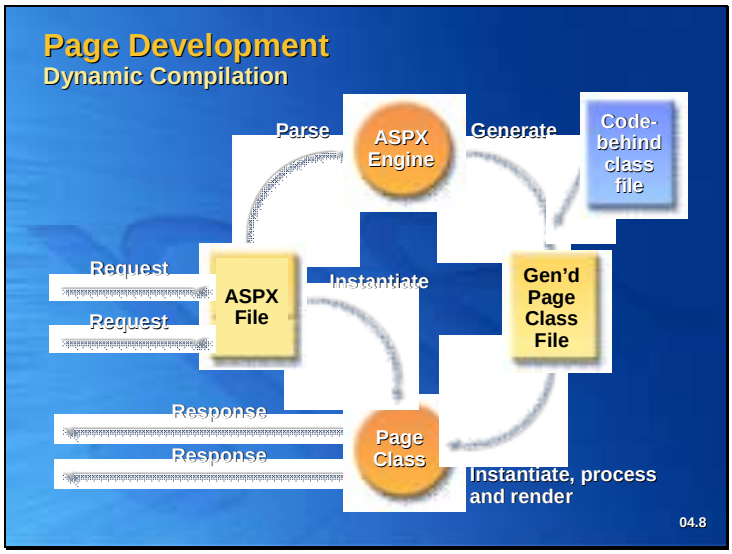
Page Development

Separation of Code from Content

In an ASP page, declarative tags (HTML, ASP directives, server controls and static text) are combined with the code. ASP allows you to perform some code separation, but it is difficult to cleanly separate code from the tags.

In ASP.NET, your code either can be stored on the same page as the tags or on a separate page known as the code behind page. This code separation allows applications to be cleanly structured and to perform much faster than the traditional ASP.

This slide shows that code can be separated into either single pages or separate pages (code and tags).



Page Development

Dynamic Compilation

This slide shows how the ASP's CLR engine works when users visit a particular page. When the user requests a page from the browser or the application, the ASPX file is parsed by the ASPX engine. This engine generates a page class file, which is processed (compiled) by the CLR before the output is rendered. These compiled bits are persisted to (saved on) a disk. So, the next time a user visits the page, the compiled bits are picked up from the disk and executed. The file is not recompiled unless a change occurs.

ASP.NET – ASPX code-behind file gets compiled into a single DLL. This is an issue for team development: Different developers building pages for the same DLL. Use synchronization regularly, local working is more effective in a team effort. Then all developers build to the same Web. The DLL is a strongly typed object.

The CLR environment requires less coding on your part because Visual Studio .NET pre-fills more of the basics into your ASP.NET Web pages when you create your project. The separation of programming code from HTML tags and presentation content gives a better view of your programming.

 ©2002 Microsoft Corporation

Page Development
Server Controls
Server-programmable objects
Properties, method and events
Encapsulate behavior
Browser-neutral - Can render differently to support multiple browsers or other Web clients
You can create custom server controls
04.9

Page Development

Server Controls

ASP.NET server controls are components that run on the server and encapsulate user interface and other related functionality. They are used in ASP.NET pages and in ASP.NET code-behind classes. This section contains background information and examples to help you develop your own ASP.NET server controls.

Note: The terms ASP.NET page and Web Forms page are synonymous and are used interchangeably in the documentation.

The server controls are programmable in that they provide properties, methods and events for use in your applications. The controls are programmable in your server code, allowing you to interact with those controls and their data. For instance, a button server control may provide code that accesses the properties in the text boxes on a page. Your code only needs to interact with the controls properties to access this data.

For instance, to access a text box you could use this code:

```
MyName = txtMyName.Text
```

Many of the controls are databound so you can quickly link data to a control. For instance, the DataGrid can be bound to both ADO.NET data and other data such as XML data.

Specific Internet Explorer 5.5 WebControls

Microsoft provides a set of server controls for IE 5.5 and higher. The Microsoft® Internet Explorer WebControls are a powerful collection of ASP.NET server controls. The WebControls implement a single-source authoring solution for four popular user-interface (UI) elements: MultiPage, TabStrip, Toolbar and TreeView.

Up-level browsers are a Beta 1 concept including IE 5.0 and higher browsers. In Beta 2, the browsecaps listings for each browser are stored in the machine.config file. HTML TextWriter classes determine how the HTML stream is rendered for a particular browser, for example, whether a tag is rendered as a

 or some other tag. For more information on the TextWriter class, please see http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vjref98/html/com.ms.wfc.io.TextWriter.asp.

The Internet Explorer WebControls deliver content useful in all common browsers while taking advantage of powerful features supported by IE 5.5 and higher. The Internet Explorer WebControls provide an authoring solution with widespread reach, by delivering HTML 3.2 compatible content to down-level browsers. ASP.NET Web Forms detect the client browser capabilities and include DHTML behaviors in the Web pages downloaded to up-level browsers. The objects exposed by the DHTML behaviors and the ASP.NET server controls are presented in the Internet Explorer WebControls References.

Custom Server Controls

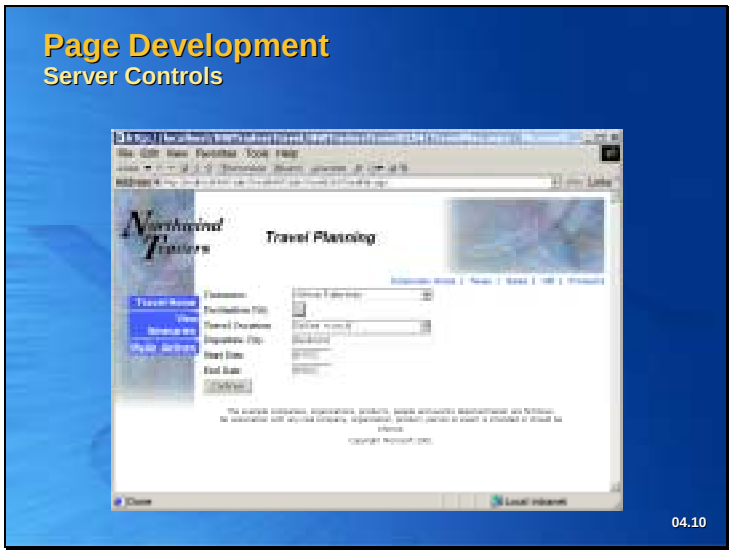
You can create custom server controls with the Graphical User Interface features and classes found in Visual Studio .NET and the .NET Framework. Creating these is similar to creating the controls in Windows Forms and support for the shared functionality (custom value editors, designer verbs, etc.) is described in the Programming Guide of the .NET Frameworks documentation. The Programming Guide describes the design-time support that is unique to ASP.NET Server Control authoring.

Server controls are executed on the server and the upside is that they provide browser-neutral code for the client. Event processing of server controls happens on the server, the event model is distinctly similar to Visual Basic and the output from the server rendered to the HTML level specified in the application, so it can be browser-neutral or browser-specific. With the Mobile Internet toolkit, the output can be rendered to fit the requirements of the user device as well.

The downside is that server controls require more round trips from client to server and back to client, but if your application must support all browsers, then server controls are the answer. With ASP.NET and Visual Studio .NET, you have plenty of options for when and how to use Client Script or server controls. If your application requires browser-neutral code, then it is worth the time it will take to make the round trip to the server and back. However, if your application does not have to be completely browser-neutral, you can reduce round trips with browser-specific code. Cross-browser support works well in Visual Studio .NET. There are specific controls for Internet Explorer that can perform exciting new tasks such as creating menu trees and dynamic lists. You can even package your code into Client Controls. More on this topic is to come later in the training session.

All ASP.NET pages (.aspx pages) are compiled into DLLs and executed as native code regardless of the language in which they were created.

 ©2002 Microsoft Corporation



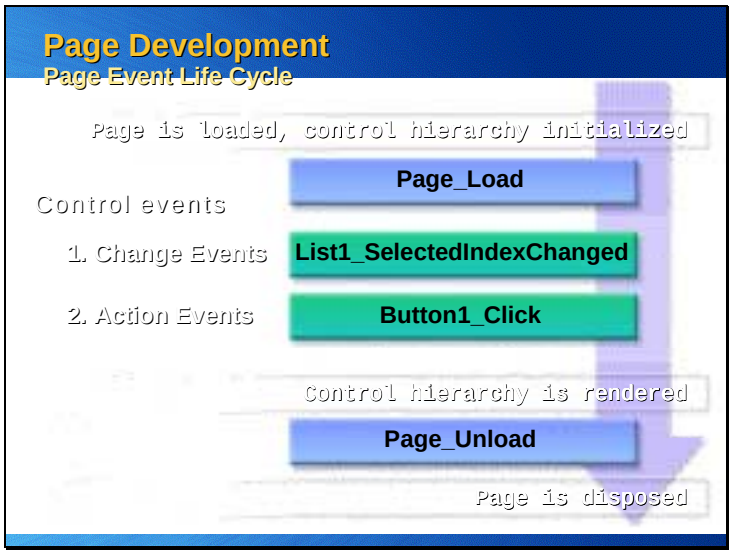
Page Development

Server Controls

This slide illustrates a simple page built with server controls that uses text boxes, buttons, images and calendar controls. Each control can be accessed in server code to reduce the amount of coding necessary. For instance, the calendar is rendered with the Calendar Web server control, though there is no HTML or server coding that the developer must do.

A complete listing of Web server controls is available at:

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpgenref/html/cpconaspsyntaxforwebcontrols.asp



Page Development

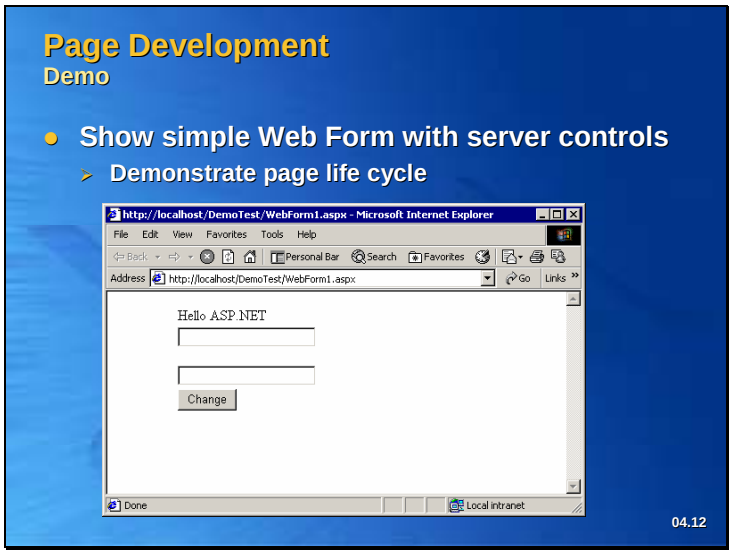
Page Event Life Cycle

When a user visits an ASPX page, it demands a particular cycle of event processing. It is important to understand that this event-processing cycle occurs on the server between the time the user visits the page and the time the resulting HTML is sent to the browser. The event-processing cycle implements an event-processing system similar to Visual Basic.

When a page is requested, and the controls are initialized: the **Page_Load** event fires. You can use the Page.IsPostBack property to determine when a Page_Load occurs during a post-back operation from a server control. Once the Page_Load occurs, the other events occur in the sequence shown on the slides.

- Control action events
- Button_Click
- DataList_ItemCommand
- Page_Unload
- Fires at end of a request

 ©2002 Microsoft Corporation



Page Development

Demo

The purpose of this demo is to illustrate:

- The ease of creating Web Forms
- Power of server controls
- Event paradigm of pages and controls
- Standard HTML generation of controls

Show simple Web Forms with server controls and demonstrate the Page Event Life Cycle.

Page Development

Input Validation

Data validation is one of the most serious problems with application development. Traditionally, developers had to create their own heavily code-based validation schemes. For instance, you were forced to use IF statements or another testing technique to test user input at a number of places.

To make validation in Web Applications easier, ASP.NET provides a set of controls to handle field validation. There are a series of server controls you can use to validate input on your Web Forms pages and there are controls for different types of validation, such as range checking or pattern matching. These controls appear on the Web Forms tab of the Toolbox.

To use the Validation Controls, simply drag a Validation Control from the Toolbox and drop it on the form, then set the errormessage and controltovalidate properties. You also may need to set other properties, set ranges or other parameters the Validator checks. These steps wire the Validation Control to your control.

The Validation Controls are:

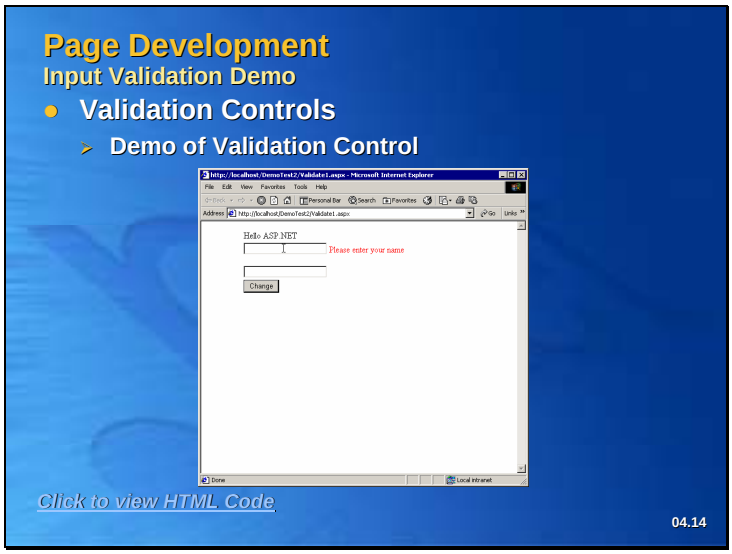
- **RequiredFieldValidator** - Used to ensure the user enters all required fields when entering data on a Web Forms page. For more information, see Validating Required Entries for Web Server Controls
- **CompareValidator** - Used to compare a user's entry against a constant value, against the property value of another control or against a database value. For more information, see Validating Against a Specific Value for Web Server Controls or Validating Against a Data Type for Web Server Controls
- **RangeValidator** - Used to check that a user's entry is between specified upper and lower bounds. For more information, see Validating Against a Range of Values for Web Server Controls

 ©2002 Microsoft Corporation

- **RegularExpressionValidator** - Used to check that a user's entry matches a pattern defined by a regular expression. For more information, see Validating Against Patterns for Web Server Controls
- **CustomValidator** - Used to check that a user's entry matches the validation logic you create. For more information, see Validating with a Custom Function for Web Server Controls

If the browser that renders the validation controls understands Java script then the client side Java script is generated and executes on the client thus saving the round trip to the server. However if the client is unable to understand Java script then server side Java script is created that executes on server. It ensures that the validation code will always execute.

These validation controls generate appropriate client-side Java script is created that saves round trips to servers.



Page Development

Input Validation Demo

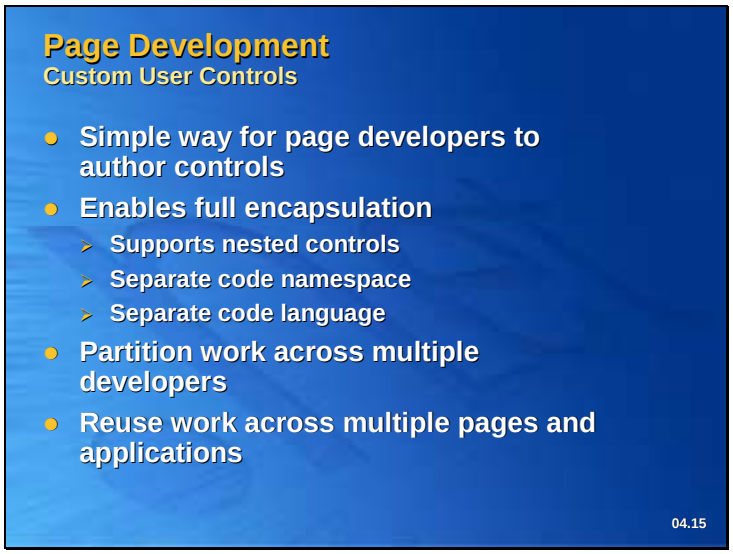
The purpose of this demo is to illustrate:

- Linking validation to controls
- Validation control types
- Declarative mechanism of controls
- Server and client logic.

 ©2002 Microsoft Corporation

Validation Controls HTML:

```
<TABLE cellSpacing="1" cellPadding="1" width="100%" border="0">
      <TR>
            <TD style="WIDTH: 64px">
            </TD>
            <TD>
                  <asp:Label id="lblMessage" runat="server">Label</asp:Label>
            </TD>
      </TR>
      <TR>
            <TD style="WIDTH: 64px">
            </TD>
            <TD>
            <P>
                  <asp:TextBox id="TextBox1" runat="server"></asp:TextBox>
            </P>
            <P>
                  <asp:TextBox id="TextBox2" runat="server"></asp:TextBox>
            </P>
            </TD>
      </TR>
      <TR>
            <TD style="WIDTH: 64px">
            </TD>
            <TD>
      <asp:Button id="cmdChange" runat="server" Text="Change"></asp:Button>
            </TD>
      </TR>
/TABLE>
```



Page Development

Custom User Controls

As a developer, you need successful ways to create custom user controls in Web Applications so you can encapsulate HTML and functionality into a control that can be reused. For instance, you can create a header control that contains the entire header for your applications so that those applications can reuse the header, and you only have to maintain one copy.

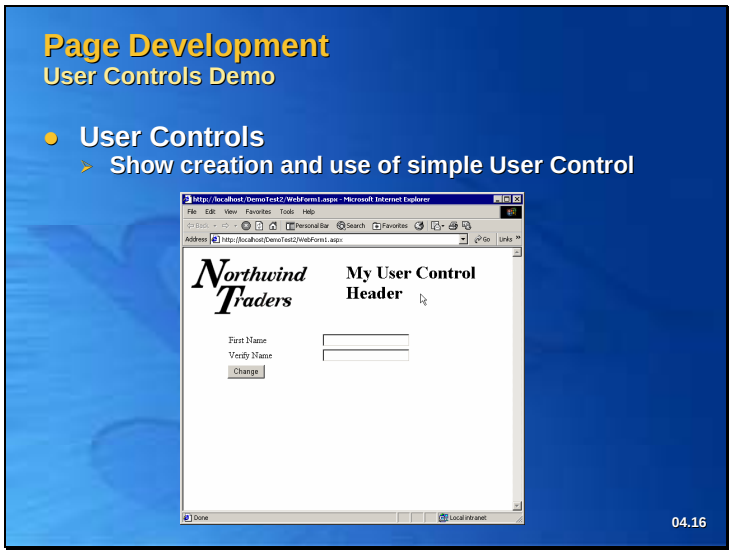
Custom controls offer you an easy way to partition and reuse common UI functionality across your ASP.NET Web Applications. Like a Web Forms page, you can author these controls with any text editor or develop them using code-behind classes. Also, like a Web Forms page, custom controls are compiled when first requested and stored in server memory to reduce the response time for subsequent requests. Unlike pages, however, custom controls cannot be requested independently. They must be included in a Web Forms page.

Creating your own controls also provides greater flexibility than server-side includes (SSIs) by accessing the object model support provided by ASP.NET. Rather than simply including the functionality provided by another file, you can program against any properties you declare in the control, just like any other ASP.NET server control.

Controls can contain other controls, images or any valid HTML. You can also use code-behind files with application logic in your controls. For instance, in a header control you can place a property that allows you to set the header for the application.

In addition, custom controls have logic that is able to access a database or other resource and even display information from those resources.

 ©2002 Microsoft Corporation



Page Development

User Controls Demo

User Controls

The purpose of this demo is to illustrate the ease of creating controls from standard HTML; standardization using controls; integration of controls onto pages using Visual Studio .NET; and show how to add properties to user controls.

Application Services

These topics are discussed in detail on the following slides.

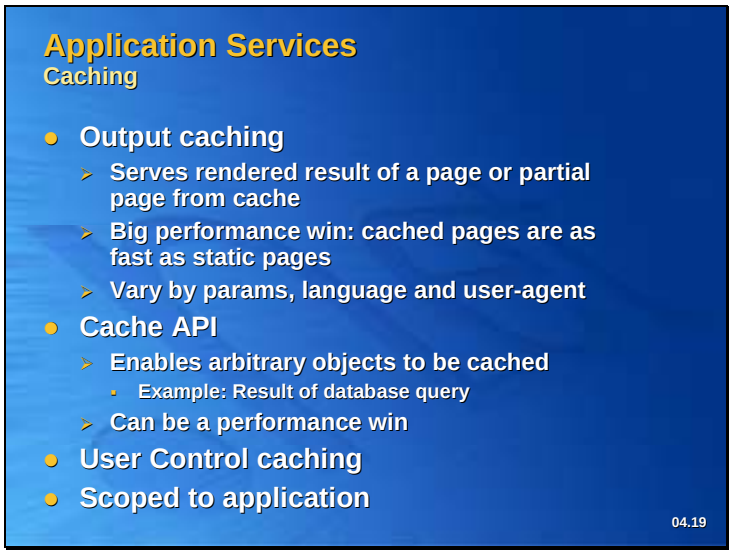
 ©2002 Microsoft Corporation

Application Services

Request Processing

ASP.NET contains a complete request processing system for Web Applications that is modular and pluggable, which means you can change the components.

The request processing services also provide features such as caching, security and session state. Having these services built-in allows you to focus on developing your core application without having to worry about creating the services provided by ASP.NET.


Application Services
Caching
Output caching
Serves rendered result of a page or partial page from cache
Big performance win: cached pages are as fast as static pages
Vary by params, language and user-agent
Cache API
Enables arbitrary objects to be cached
Example: Result of database query
Can be a performance win
User Control caching
Scoped to application
04.19

Application Services

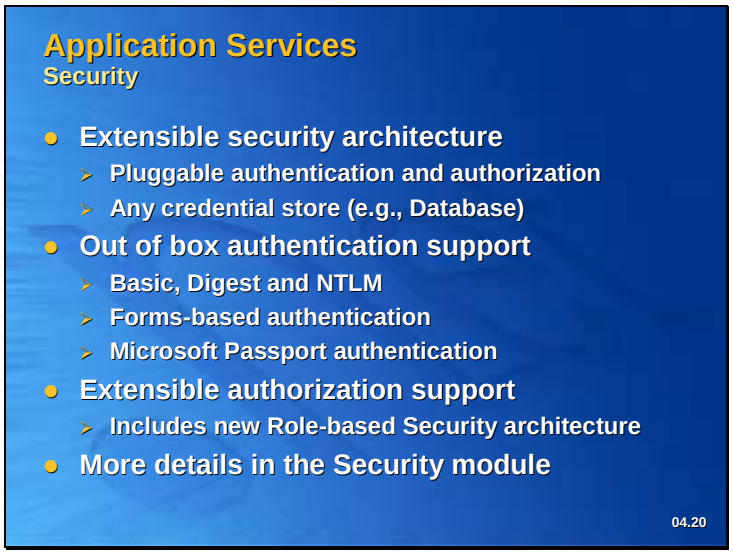
Caching

Caching is the process of storing frequently-used, and often, costly-to-generate data for reuse. Typically, this data is stored in memory since retrieving data from memory is more efficient than retrieving the data from other locations such as a database. ASP.NET has Server facilities for supporting caching: a cache API for storing arbitrary data and an @OutputCache directive used to store frequently requested ASP.NET pages. Caching services are often required in applications to improve their performance.

The ASP.NET Page @OutputCaching directive allows the contents of a given page to be stored in the cache. There are three @OutputCache directive attributes that allow you to vary how a page's output is cached. The required VaryByParam attribute allows you to vary the output cache depending on a **GET** query string or form **POST** parameters. The VaryByHeader attribute allows you to vary the cached output depending on the HTTP header associated with the request while the VaryByCustom attribute allows you to vary caching by browser type. You can also use it as an extensibility mechanism for custom cache variation.

Custom user controls also can be cached, which allows for partial page caching.
Caching can be controlled by many factors. For instance, pages can be cached and dynamically updated based upon factors such as a change in a query string. This allows dynamic pages to be cached and then updated as needed.

 ©2002 Microsoft Corporation



Application Services

Security

ASP.NET provides a powerful and extensible security system for Web Applications that enables you to integrate with existing systems or create your own security system.

With most Web sites, it is necessary to selectively restrict access to some portions of the site. You can think of a Web site as analogous to an art gallery. The gallery is open for the public to browse, but parts of the facility, such as the business offices, are restricted to people with certain credentials. When a Web site records users' credit card information, for example, the file or database that stores this information must be secured from public access. ASP.NET security features address these and many other security issues.

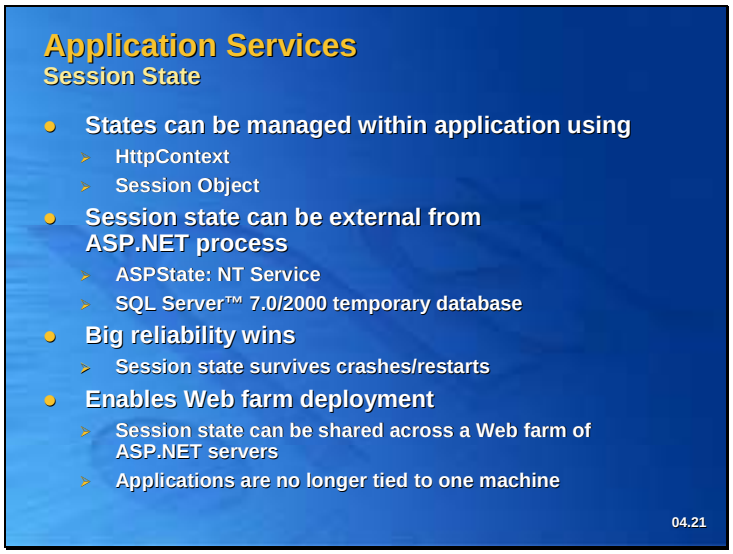
ASP.NET, in conjunction with Internet Information Services (IIS), can authenticate user credentials, such as names and passwords, using any of the following authentication methods:

- Windows: Basic, Digest or Integrated Windows Authentication (NTLM or Kerberos).
- Microsoft Passport authentication
- Forms-based authentication
- Client Certificates

ASP.NET controls access to site information by comparing authenticated credentials, or representations of them, to NTFS file system permissions or to an XML file that lists authorized users, authorized roles (groups) or authorized HTTP verbs.

The architecture also provides Role-based Security to allow you to make authorization decisions based upon those users' membership in various roles. This makes it easy to create an application that exposes certain things to particular users and other things to all users.

Security is discussed in more detail in Module 11: Security in .NET.


Application Services
Session State
States can be managed within application using
HttpContext
Session Object
Session state can be external from ASP.NET process
ASPState: NT Service
SQL Server™ 7.0/2000 temporary database
Big reliability wins
Session state survives crashes/restarts
Enables Web farm deployment
Session state can be shared across a Web farm of ASP.NET servers
Applications are no longer tied to one machine
04.21

Application Services

Session State

Session state management is a key element of Web Application design. ASP.NET contains a new session state support service that is flexible and scalable.

ASP.NET provides an infrastructure with built-in session state functionality that enables you to:

- Automatically identify and classify requests coming from a single browser client into a logical application session on the server.
- Store session-scoped data on the server for use across multiple browser requests.
- Raise appropriate session-lifetime management events (Session_OnStart, Session_OnEnd and so on) that can be handled in application code.
- Automatically release session data if the browser does not revisit an application within a specified time-out period.

ASP.NET provides the following support for sessions:

- A session-state facility that is easy to use, familiar to ASP developers and consistent with other .NET Framework APIs.
- A reliable session-state facility that can survive IIS restarts and worker-process restarts without losing session data.
- A scalable session-state facility that can be used in both Web farm (multi-computer) and Web garden (multi-process) scenarios and that enables administrators to allocate more processors to a Web Application to improve its scalability.
- A session-state facility that works with browsers that do not support HTTP cookies.

The resulting session state engine provides a high-performance state service that you can use in your applications. In-process state management works if the application runs on a single server. If you need to

 ©2002 Microsoft Corporation

move the session management to another system so your application can run in a Web farm, then the state service or SQL Server can provide the session state support. It is your choice and you do not have to do any coding to make it work.

Session state packaging issue - The premium edition of Visual Studio® provides Web farm session state and caching. Download it from the Microsoft® Developers Network by typing in www.asp.net, which has a link to the download.

Application Services

Session State Demo

The purpose of this demo is to use session state in the application to illustrate:

- Session options in Web.Config
- How state management works similar to ASP
- How View State in managed
- Discuss improvements for all types of applications

Session State WebForm1 cmdNext Code

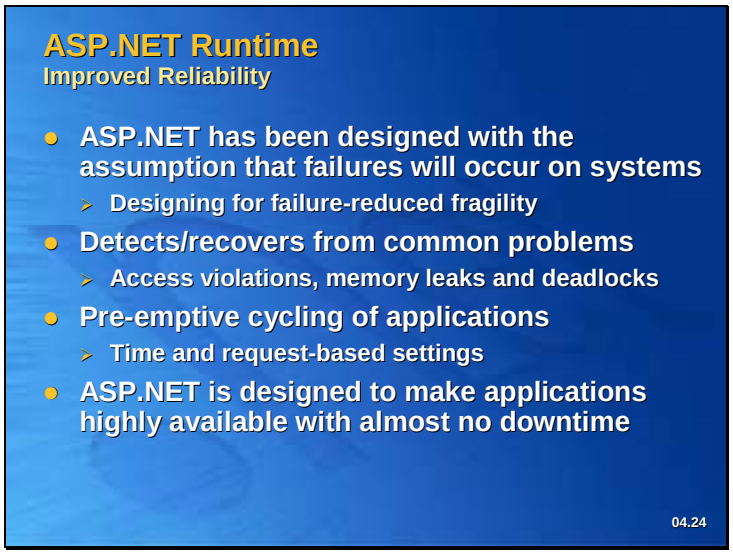
```
Session("YourName") = TextBox1.Text
Response.Redirect("ValidateIt.aspx")
```

```
Session State ValidateIt cmdNext Code
TextBox1.Text = Session("YourName")
```

 ©2002 Microsoft Corporation

ASP.NET Runtime Topics

ASP.NET improves deployment, scalability, security and reliability. With simple ASP applications, deployment isn't an issue, but you run into problems when you start moving to proper n-tier designs that utilize components. The whole process of DLL conflict (component registration, versioning locked DLLs and so on) comes into play when you have to deploy and maintain these types of applications. ASP.NET addresses this by eliminating component registration, DLL locking and the use of XML configuration files. This allows you to deploy a Web Application simply by copying directories.


ASP.NET Runtime
Improved Reliability
ASP.NET has been designed with the assumption that failures will occur on systems
Designing for failure-reduced fragility
Detects/recovers from common problems
Access violations, memory leaks and deadlocks
Pre-emptive cycling of applications
Time and request-based settings
ASP.NET is designed to make applications highly available with almost no downtime
04.24

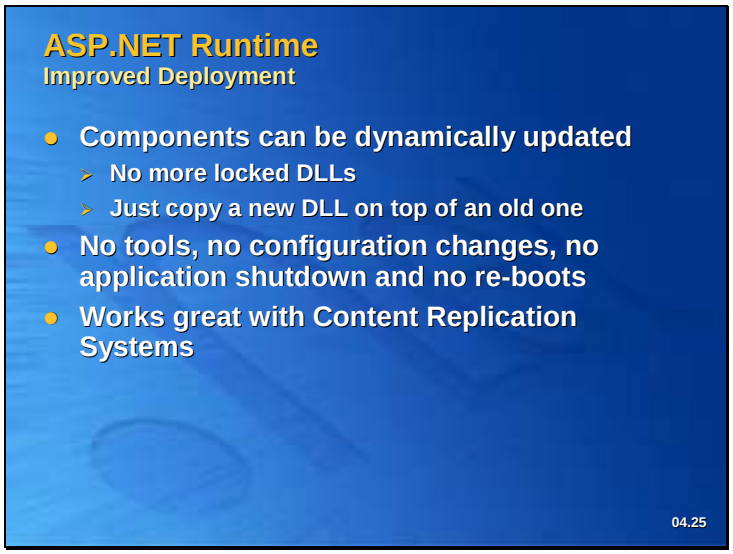
ASP.NET Runtime

Improved Reliability

ASP.NET provides a reliable CLR environment for your applications. That Runtime has been designed to take into account application scenarios where you need to restart the server periodically. For instance, you might want to restart the server once per week to make sure there are no resource issues, or you might want the server to restart based upon memory leaks in applications. You can control these settings by simply changing the application's configuration files.

This pre-emptive cycling of applications is a powerful way to make sure problem applications are restarted frequently.

 ©2002 Microsoft Corporation


ASP.NET Runtime
Improved Deployment
Components can be dynamically updated
No more locked DLLs
Just copy a new DLL on top of an old one
No tools, no configuration changes, no application shutdown and no re-boots
Works great with Content Replication Systems
04.25

ASP.NET Runtime

Improved Deployment

It is easier to employ ASP.NET applications than other application types or past Web Applications.

The .NET Framework provides the following options for distributing applications:

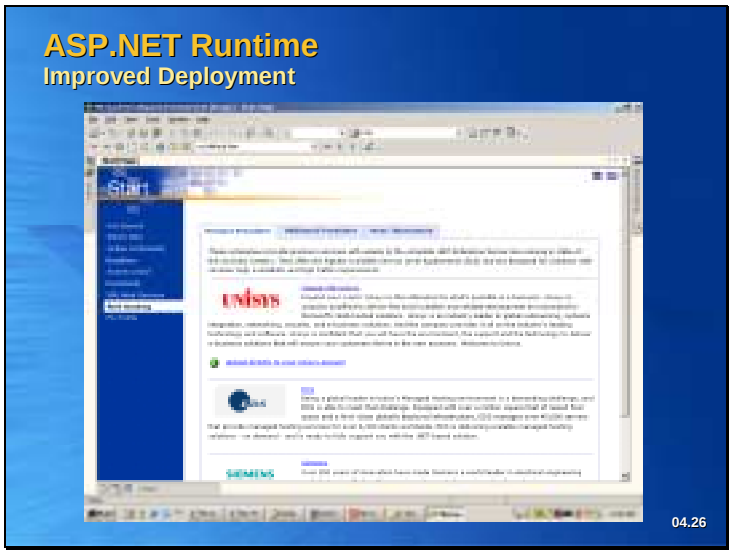
Use XCOPY or FTP – Because CLR applications are self-describing and require no registry entries, you can use XCOPY or FTP to simply copy the application to an appropriate directory. Then the application can be run from that directory.

Use code download - If you are distributing your application over the Internet or through a corporate Intranet, you can download the code to a computer and then run the application.

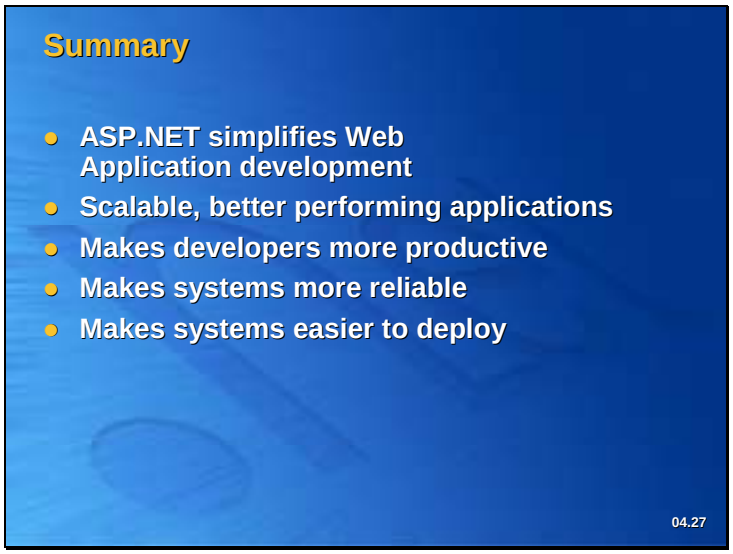
Use an installer program such as the Windows Installer 2.0 - The Windows Installer 2.0 can install, repair or remove Microsoft .NET Framework assemblies in the global assembly cache and in private directories.

Content Replication Systems also can deploy ASP.NET solutions. Since there is no need to register components or elements of applications and because the configuration files for the application go along with the application, then the CRS can move these files. However, after you XCOPY an application to its new location, you must set an Application Root for it through the Internet Services Manager so that the IIS server will recognize it.

Deploying an ASP.NET application involves packaging the application and deploying it. You can use Xcopy to deploy an ASP.NET application on to a server and then run one version of the application side-by-side without DLL conflicts. In addition, you can update the application without having to close it.

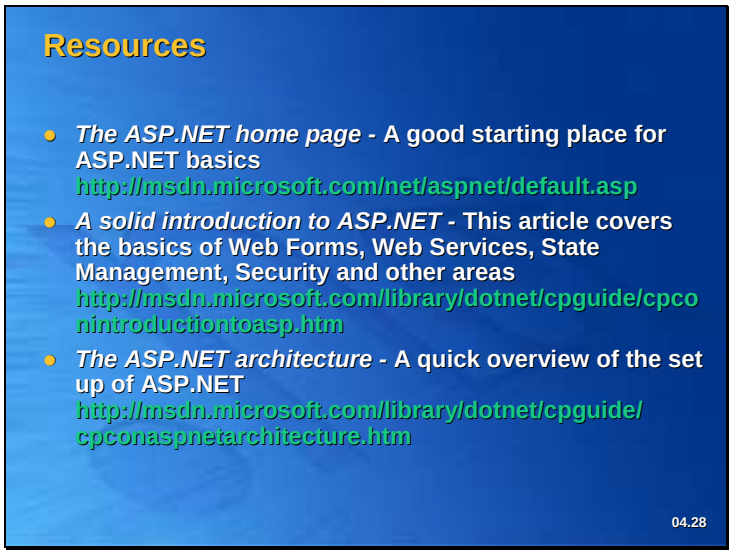


 ©2002 Microsoft Corporation



Summary

Active Server Pages long has been the foundation for creating rich and dynamic Web sites using server-side scripting. With the release of the .NET Framework, ASP has evolved into ASP.NET and now embodies many of the key concepts behind the .NET Framework. In addition to being able to access any of the programmatic interfaces exposed by the .NET Framework, you can construct server-side code using any language that is compatible with the .NET Framework.


Resources
The ASP.NET home page - A good starting place for ASP.NET basics
http://msdn.microsoft.com/net/aspnet/default.asp
A solid introduction to ASP.NET - This article covers the basics of Web Forms, Web Services, State Management, Security and other areas
http://msdn.microsoft.com/library/dotnet/cpguide/cpconintroductiontoasp.htm
The ASP.NET architecture - A quick overview of the set up of ASP.NET
http://msdn.microsoft.com/library/dotnet/cpguide/cpconaspnetarchitecture.htm
04.28

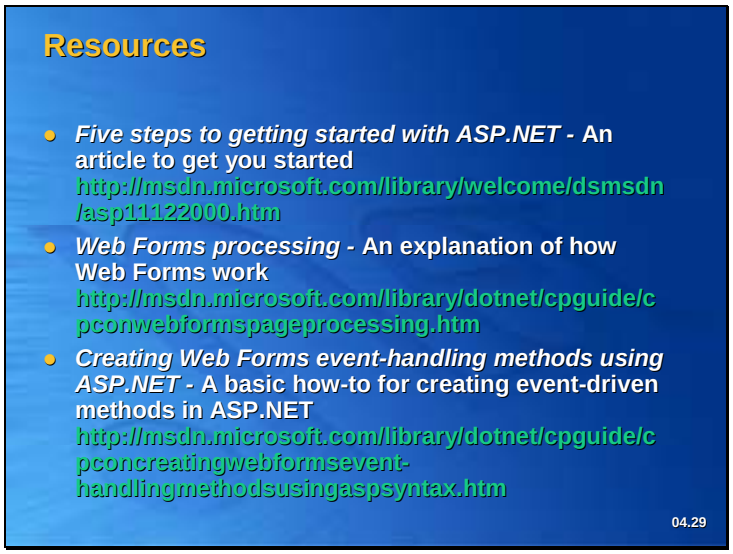
Resources

The ASP.NET home page - A good starting place for ASP.NET basics
http://msdn.microsoft.com/net/aspnet/default.asp

A solid introduction to ASP.NET - This article covers the basics of Web Forms, Web Services, State Management, Security and other areas http://msdn.microsoft.com/library/dotnet/cpguide/cpconintroductiontoasp.htm

The ASP.NET architecture - A quick overview of the set up of ASP.NET
http://msdn.microsoft.com/library/dotnet/cpguide/
cpconaspnetarchitecture.htm

 ©2002 Microsoft Corporation

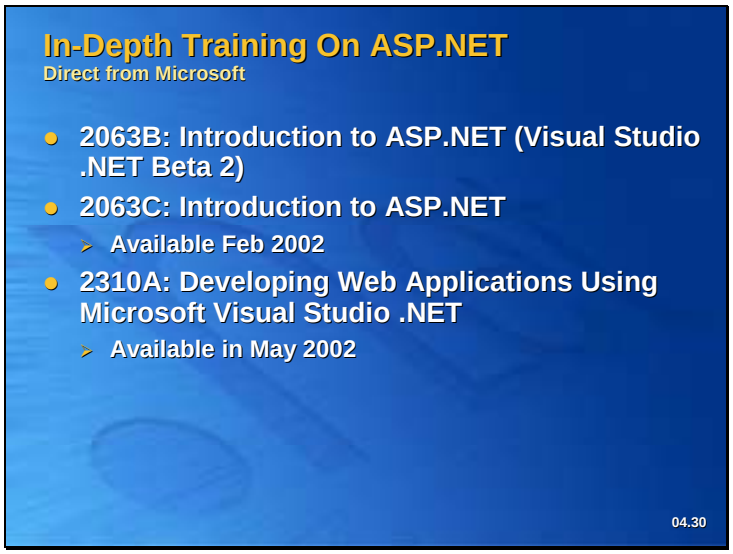

Resources
Five steps to getting started with ASP.NET - An article to get you started
http://msdn.microsoft.com/library/welcome/dsmsdn/asp11122000.htm
Web Forms processing - An explanation of how Web Forms work
http://msdn.microsoft.com/library/dotnet/cpguide/cpconwebformspageprocessing.htm
Creating Web Forms event-handling methods using ASP.NET - A basic how-to for creating event-driven methods in ASP.NET
http://msdn.microsoft.com/library/dotnet/cpguide/cpconcreatingwebformsevent-handlingmethodsusingaspsyntax.htm
04.29

Resources

Five steps to getting started with ASP.NET - An article to get you started
http://msdn.microsoft.com/library/welcome/dsmsdn/asp11122000.htm

Web Forms processing - An explanation of how Web Forms work
http://msdn.microsoft.com/library/dotnet/cpguide/cpconwebformspageprocessing.htm

Creating Web Forms event-handling methods using ASP.NET - A basic how-to for creating event-driven methods in ASP.NET http://msdn.microsoft.com/library/dotnet/cpguide/cpconcreatingwebformsevent-handlingmethodsusingaspsyntax.htm


In-Depth Training On ASP.NET
Direct from Microsoft
2063B: Introduction to ASP.NET (Visual Studio .NET Beta 2)
2063C: Introduction to ASP.NET
Available Feb 2002
2310A: Developing Web Applications Using Microsoft Visual Studio .NET
Available in May 2002
04.30

In-Depth Training On ASP.NET

Direct from Microsoft

 ©2002 Microsoft Corporation

Microsoft
04.31